Operating System

Plug and Play for Windows 2000

White Paper

Abstract

This paper describes the Microsoft® Windows® 2000 operating system implementation of Plug and Play. Plug and Play is one of a number of enhancements to Windows that will simplify device driver development and device management.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

0499

CONTENTS

INTRODUCTION

The Microsoft® Window® 2000 operating system includes enhancements to simplify device driver development and device management. These enhancements include support for low-level instrumentation, power management, and Plug and Play (the subject of this white paper).

Plug and Play is a combination of hardware and software support that enables a computer system to recognize and adapt to hardware configuration changes with little or no user intervention. With Plug and Play, a user can add or remove devices dynamically, without awkward and confusing manual configuration and without any intricate knowledge of computer hardware. For example, a user can dock a portable computer and use the docking station's Ethernet card to connect to the network without changing the configuration. Later, the user can undock that same computer and use a modem to connect to the network—again without making any manual configuration changes.

Plug and Play allows a user to change a computer's configuration with the assurance that all devices will work together and that the machine will boot correctly after the changes are made.

The Evolution of Plug and Play

Support for Plug and Play was first provided in the Microsoft Windows® 95 operating system; however, since that time Plug and Play has evolved dramatically. This evolution is largely a result of the OnNow design initiative, which seeks to define a comprehensive, system-wide approach to controlling system and device configuration and power management. One product of the OnNow initiative is the *Advanced Configuration and Power Interface* (ACPI) *version 1.0* specification, which defines a new system board and BIOS interface that extends Plug and Play data to include power management and other new configuration capabilities, all under complete control of the operating system.

Unlike Plug and Play support in Microsoft Windows 95, the Windows 2000 Plug and Play implementation does not rely on an Advanced Power Management (APM) BIOS or a Plug and Play BIOS. These two BIOS implementations were designed for Windows 95 as early attempts to support Plug and Play and power management; they will be maintained in Windows 98 for backward compatibility only. ACPI provides these services for both Windows 98 and Windows 2000.

The primary design goal of Plug and Play is to further the industry initiative to simplify personal computers for end users. Beyond that, Windows 2000 Plug and Play is designed to:

- Extend the existing Windows I/O infrastructure to support Plug and Play and power management, while also supporting industry hardware standards for Plug and Play.
- Achieve common device driver interfaces that support Plug and Play and power management for many device classes under Windows 2000 and Windows 98.

In Windows 2000, Plug and Play support is optimized for laptop, workstation, and

server computers that include ACPI system boards. In addition, Plug and Play device driver support for many device classes is provided by the Microsoft Win32® Driver Model (WDM), which also supports power management and other new capabilities that can be configured and controlled by the operating system.

The Windows 2000 Implementation

To incorporate Plug and Play support into Windows 2000, a native Plug and Play implementation was integrated into the existing Windows code base. This results in the following changes for developers who previously created drivers under the Windows NT 4.0 device driver model:

- Bus drivers are now separate from the HAL. Bus drivers control an I/O bus, including per-slot functionality that is device-independent. In the new architecture, bus drivers have moved out of the hardware abstraction layer (HAL) to coordinate with changes and extensions made to existing kernel-mode components, such as the Executive, device drivers, and the HAL. Bus drivers are generally provided by Microsoft.
- New methods and capabilities are available to support device installation and configuration. The new design includes changes and extensions to existing user-mode components, such as the Spooler, class installers, Control Panel applications, and Setup. In addition, new kernel-mode and user-mode Plug and Play–enabled components have been added.
- New Plug and Play APIs are used to read and write information from the registry. For the new design, changes and extensions were made to the registry structure. This structure supports Plug and Play and allows the registry to be enhanced in future versions of Windows, while also providing backward compatibility.

Windows 2000 supports legacy Windows NT drivers, but these have no Plug and Play and power management functionality. Manufacturers who want to support complete Plug and Play capabilities for their devices and who want the same drivers to function on both Windows NT and Windows operating systems need to develop new drivers that integrate the latest Plug and Play and power management functionality.

This white paper provides a brief overview of the architecture and design direction for Plug and Play under Windows 2000. The beta releases of the Windows 2000 DDK document the actual driver modifications required to allow current drivers to work in a Windows 2000 Plug and Play system.

PLUG AND PLAY OVERVIEW

A Plug and Play system requires the combined interaction of the personal computer's BIOS, hardware components, device drivers, and operating system software. The basic system board implementation and BIOS support required for Plug and Play support under Windows 2000 are defined in the ACPI specification. Both Windows 2000 and Windows 98 use this specification as the basis for their Plug and Play and OnNow architecture.

The ACPI specification defines a new interface between the operating system and the personal computer's Plug and Play and power management features. The ACPI methods defined are independent of the actual operating system or CPU. ACPI specifies a register-level interface to core Plug and Play and power management functions and defines a descriptive interface for additional hardware features. This gives system designers the ability to implement a range of Plug and Play and power management features with different hardware designs while using the same operating-system driver. ACPI also provides a generic system-event mechanism for Plug and Play and power management.

In addition to the ACPI specification, other hardware support is defined in industry standards, such as *Universal Serial Bus, Version 1.0*, *PCI Local Bus Specification, Revision 2.1*, or PCMCIA standards, and in the Plug and Play specifications. (See the References section at the end of this paper for locations of all specifications.)

System Support for Plug and Play

Windows 2000 provides the following support for Plug and Play:

- **Automatic and dynamic recognition of installed hardware.** This includes initial system installation, recognition of Plug and Play hardware changes that occur between system boots, and response to run-time hardware events, such as dock or undock and device insertion or removal.
- **Hardware resource allocation (and reallocation).** Drivers for Plug and Play devices do not assign their own resources. Instead, the required resources for a device are identified when the device is enumerated by the operating system. The Plug and Play Manager retrieves the requirements for each device during resource allocation. Based on the resource requests that each device makes, the Plug and Play Manager assigns the appropriate hardware resources, such as I/O ports, IRQs, DMA channels, and memory locations. The Plug and Play Manager reconfigures resource assignments when needed, such as when a device is added to the system, and requests resources that are already in use.
- **Loading of appropriate drivers.** The Plug and Play Manager determines which drivers are required to support a particular device and loads those drivers.
- **An interface for driver interaction with the Plug and Play system.** The interface consists primarily of I/O routines, Plug and Play I/O request packets (IRPs), required driver entry points, and information in the registry.
- **Interaction with power management.** One of the key features of both Plug and Play and power management is dynamic handling of events. The addition

or removal of a device is an example of such a dynamic event, as is the ability to awaken a device or put it to sleep. Plug and Play and power management both use WDM-based functions and have similar methods for responding to dynamic events.

- **Registration for device notification events.** Plug and Play enables user-mode code to register for and be notified of certain Plug and Play events. The **RegisterDeviceNotification** routine allows callers to filter exactly the class or device for which they want to receive notification. This can be specific, such as a file system handle, or general, such as a class of devices. Legacy Windows NT notification methods continue to work as before.

Device and Driver Support Levels

The extent to which a device supports Plug and Play depends on the Plug and Play support in both the device hardware and the driver(s) for the device (see Table 1).

	Plug and Play driver	**Non-Plug and Play driver**
Plug and Play device	Full Plug and Play	No Plug and Play
Non-Plug and Play device	Possible partial Plug and Play	No Plug and Play

As this table indicates, any device that supports Plug and Play should have Plug and Play support in its driver. The following list expands on the possible configurations:

- **Plug and Play device and driver—full Plug and Play support.** To provide the optimal Plug and Play support, the hardware implementation must comply with the OnNow design initiative, including ACPI. Windows 2000 Plug and Play targets ACPI systems only.
- **Plug and Play device/non-Plug and Play driver—no Plug and Play support.** If a driver does not support Plug and Play, its device behaves as a non-Plug and Play device, regardless of support in the hardware. A non-Plug and Play device can constrain the Plug and Play capabilities of the entire system.
- **Non-Plug and Play device/Plug and Play driver—possible partial Plug and Play support.** A non-Plug and Play device can have partial Plug and Play support. Although it is not possible for the system to automatically and dynamically recognize the hardware and load the appropriate drivers, it is possible to have Plug and Play manage resource allocation, provide an interface for driver interaction with the Plug and Play system, interact with power management, and register device notification events. Also, if a non-Plug and Play device has a Plug and Play driver, the device appears in the Device Manager application, and property pages are available for that device.
- **Non-Plug and Play device/non-Plug and Play driver: no Plug and Play support.**

Legacy drivers written before Plug and Play was incorporated into the operating system continue to function as they did previously (without any Plug and Play capability). All new drivers should support Plug and Play.

WINDOWS 2000 PLUG AND PLAY ARCHITECTURE

Kernel-mode functionality in Windows 2000 Plug and Play supports boot-time Plug and Play activity and interfaces with the HAL, Executive, and device drivers. User-mode functionality cooperates with kernel-mode components to provide dynamic configuration and interfaces with other components that need to participate in Plug and Play, such as Setup and Control Panel.

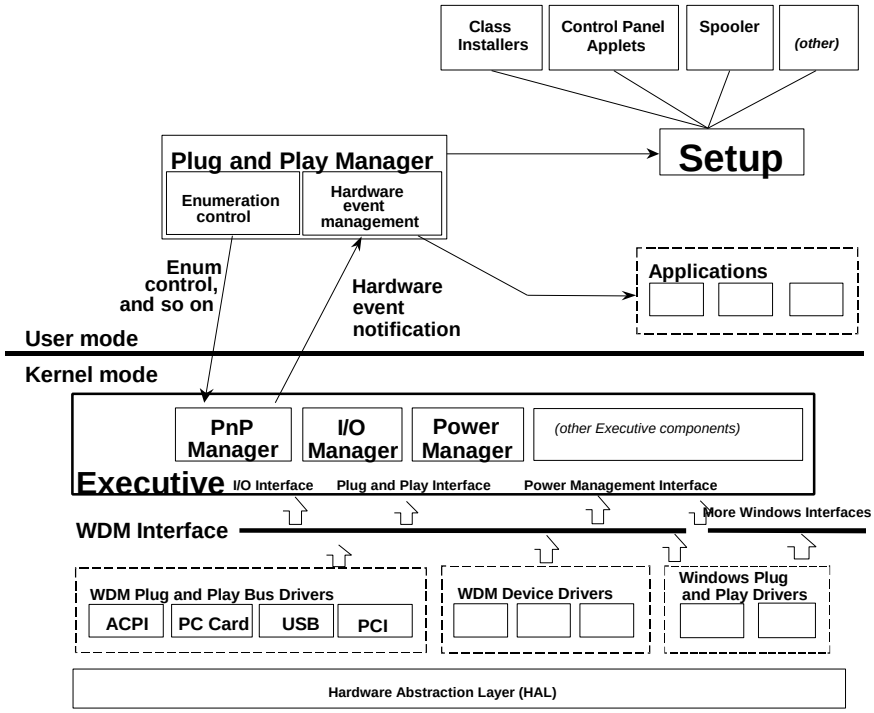

Figure 1. Windows 2000 Plug and Play architecture

Plug and Play modules shown in Figure 1 are described at length in the following sections.

Kernel-mode Plug and Play Manager

The kernel-mode Plug and Play Manager maintains central control, directing bus drivers to perform enumeration and configuration and directing device drivers to add a device, start a device, and so on.

For example, the Plug and Play Manager can send requests to determine whether a device can be safely paused or removed and to give the device driver a chance to synchronize outstanding I/O requests to the incoming request. The Plug and Play Manager coordinates with the user-mode Plug and Play counterpart to pause or remove devices that are available for such actions.

Power Manager and Policy Manager

The Power Manager is the kernel-mode component that works in combination with the Policy Manager to handle power management APIs, coordinates power events, and generates power management IRPs. For example, when several devices

request to be turned off, the Power Manager collects those requests, determines which requests must be serialized, and then generates appropriate power management IRPs.

The Policy Manager monitors activity in the system and integrates user status, application status, and device driver status into power policy. Under specified circumstances or upon request, the Policy Manager generates IRPs to change device power states.

I/O Manager

The I/O Manager provides core services for device drivers. The I/O Manager is the kernel-mode component that translates user-mode read and write commands into read or write IRPs. It manages all the other main operating system IRPs. These interfaces work as they did in Windows NT 4.0. Because both Windows NT 4.0 and Windows 2000 include the I/O manager, a Plug and Play driver can be manually installed on Windows NT 4.0 and can function as a Windows 2000 Plug and Play driver.

WDM Interface for Plug and Play

The I/O system provides a layered architecture for drivers. This section discusses types of WDM drivers, driver layers, and device objects. For a different discussion of the topics covered in this section, see the Introduction to Plug and Play Help file in the Windows 98 DDK Preview.

Types of Drivers

From the Plug and Play perspective, there are three kinds of drivers:

- A *bus driver* services a bus controller, adapter, bridge, or any device that has child devices. Bus drivers are required drivers and are generally provided by Microsoft; there is one bus driver for each type of bus on a system.
- A *function driver* is the main device driver and provides the operational interface for its device. It is a required driver unless the device is used raw (an implementation in which I/O is done by the bus driver and any bus filter drivers). The function driver for a device is typically implemented as a driver/minidriver pair. In such driver pairs, a *class driver* (usually written by Microsoft) provides the functionality required by all devices of that type, and a *minidriver* (usually written by the device vendor) provides device-specific functionality. The Plug and Play Manager loads one function driver for each device.
- A *filter driver* sorts I/O requests for a bus, a device, or a class of devices. Filter drivers are optional and can exist in any number, placed above or below a function driver and above a bus driver. Usually, filter drivers are supplied by system OEMs or independent hardware vendors (IHVs).

  In most cases, lower-level filter drivers modify the behavior of device hardware. For example, a lower-level class filter driver for mouse devices could provide acceleration, performing a non-linear conversion of mouse movement data.

Upper-level filter drivers usually provide added-value features for a device. For example, an upper-level device filter driver for a keyboard could enforce additional security checks.

Driver Layers

For a given device, there are two or more driver layers: a bus driver for the underlying I/O bus (or the Plug and Play Manager for root-enumerated devices) and a function driver for the device. Optionally, one or more filter drivers can be provided for the bus or device.

Device Objects

A driver creates a device object for each device that it controls; the device object represents the device to the driver. From the Plug and Play perspective, there are three kinds of device objects: physical device objects (PDOs), functional device objects (FDOs), and filter device objects. PDOs represent a device on a bus; every Plug and Play API that refers to a device refers to the PDO. FDOs represent the functionality of a device to a function driver. Filter device objects represent a filter driver as a hook to add value. These three kinds of device objects are all of the type DEVICE_OBJECT, but are used differently and can have different device extensions.

Additional Windows Interfaces

Windows 2000 Plug and Play drivers are not limited to using the WDM interfaces. Drivers can call other interfaces to support legacy Windows NT drivers, detection, or other Windows-specific capabilities that are not provided under WDM.

Notice that a driver that supports features specific to Windows NT is no longer compatible with Windows 98. If a driver will be used under both Windows NT and Windows 98, only WDM interfaces can be used.

WDM Bus Drivers

Bus power management and Plug and Play are controlled by WDM bus drivers, which are standard WDM drivers that expose bus capabilities. In this context, any device from which other devices are enumerated is referred to as a *bus*. A bus driver responds to new Plug and Play and power management I/O request packets (IRPs) and can be extended using filter drivers.

The bus driver is primarily responsible for the following:

- Enumerating the devices on its bus.
- Reporting dynamic events on its bus to the operating system.
- Responding to Plug and Play and power management IRPs.
- Multiplexing access to the bus (for some buses).
- Generically administering the devices on its bus.

During enumeration, a bus driver identifies the devices on its bus and creates device objects for them. The method that a bus driver uses to identify connected devices depends on the particular bus.

A bus driver performs certain operations on behalf of the devices on its bus but usually does not handle reads and writes to the devices on its bus. (A device's function driver handles reads and writes to a device.) A bus driver acts as a function driver for its controller, adapter, bridge, or other device.

Microsoft provides bus drivers for most common buses, including PCI, Plug and Play ISA, SCSI, and USB. Other bus drivers can be provided by IHVs or OEMs. A bus driver can be implemented as a driver/minidriver pair, the way a SCSI port/miniport pair drives a SCSI host adapter. In such driver pairs, one driver is linked to the second driver, and the second driver is a DLL.

The ACPI driver fulfills the role of both bus driver and function driver. ACPI allows the system to learn about devices that either do not have a standard way to enumerate themselves (that is, legacy devices) or are newly defined ACPI devices to be enumerated by ACPI (such as the LID device or the embedded controller device). ACPI also installs upper-level filter drivers for devices that have functionality beyond the standard for their bus. For example, if a PCI bus driver installs a graphics controller with power controls that are not supported by the PCI bus, the device can access its added functionality if the ACPI driver loads an upper-level filter driver for it.

WDM Device Drivers

WDM device drivers are usually the function driver/minidriver pair and filter drivers discussed in the WDM Interface for Plug and Play section above. In addition to providing the operational interface for its device, function drivers play an important role in a power-managed system, contributing information as the policy owner for the device about power management capabilities and carrying out actions related to transitions between sleeping and fully on power states.

User-Mode Plug and Play Components

The Windows 2000 user-mode APIs for controlling and configuring devices in a Plug and Play environment are 32-bit extended versions of Windows 95–based Configuration Manager APIs. In Windows 95, the Configuration Manager is a virtual device driver (VxD) that exposes these routines as services to both ring 0 and ring 3 components.

In Windows 2000, these routines expose functionality from the user-mode Plug and Play Manager and are exclusively user-mode APIs. The Windows Setup program installs the drivers, and so on. The 32-bit device installer installation APIs that Setup uses to install drivers are functionally a superset of the Windows 95 SetupxDi routines.

Windows 2000 provides APIs that applications can use for customized hardware event management and to create new hardware events.

PLUG AND PLAY DEVICE TREE

The Plug and Play Manager maintains a device tree, viewable through Device Manager, which keeps track of the active devices in the system and information about those devices. The Plug and Play Manager updates the device tree as devices are added and removed or as resources are reallocated. The device tree is hierarchical, with devices on a bus represented as children of the bus adapter or controller. The registry is the central repository for static hardware information. Plug and Play system components and drivers build, maintain, and access new and existing subtrees in the registry .

During enumeration, data for each device is stored under a new **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum** key in the registry (this is the enum tree). Plug and Play makes decisions about which device drivers are loaded based on the results of enumeration. Thus, there is an important connection between the enum tree and the services list in **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services**.
Figure 2 shows a Plug and Play device tree for a hypothetical ACPI system configuration. In practice, a device tree would consist of many additional devices, but Figure 2 shows only the devices needed in this discussion.

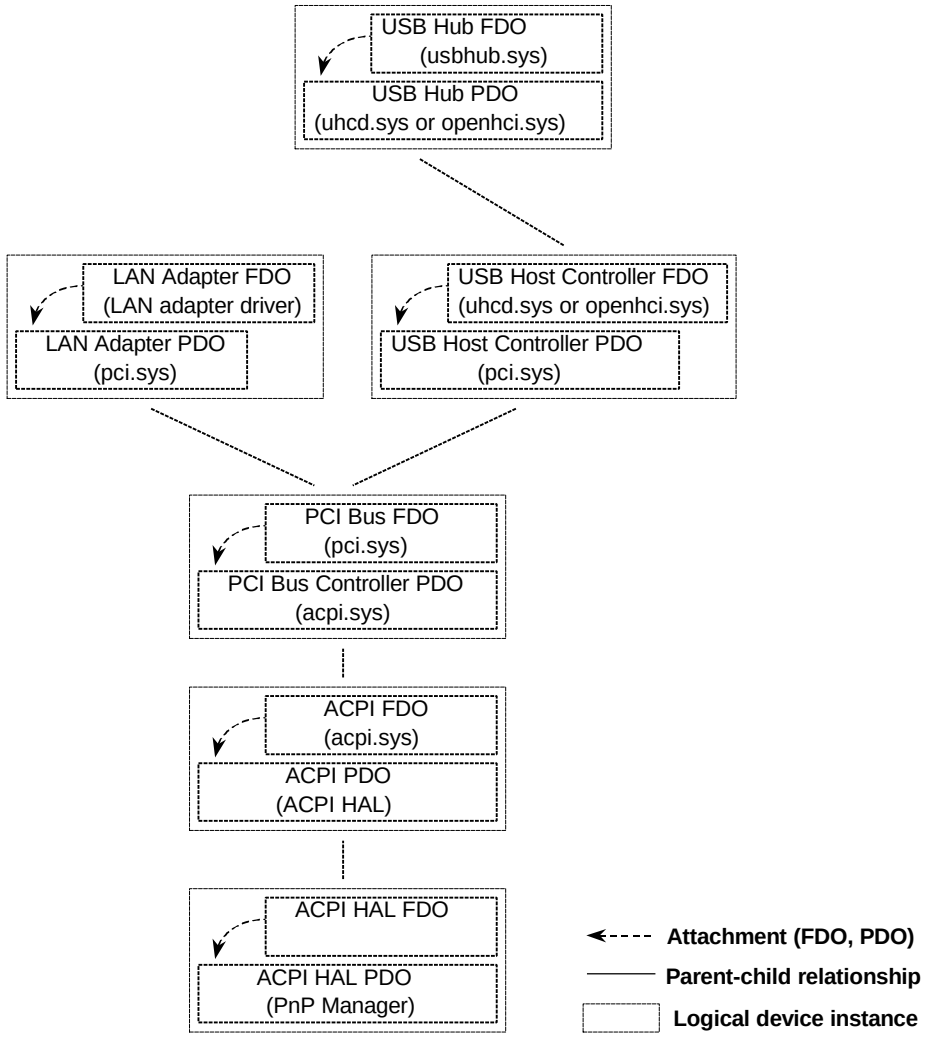

Figure 2. Sample Windows 2000 Plug and Play device tree for an ACPI system

In Figure 2, a LAN adapter and a USB host controller plug into a PCI bus. A USB hub plugs into the sole port of the USB host controller. From a Plug and Play perspective, the PCI bus controller, the USB host controller, and the USB hub are bus devices because they each provide ports. The LAN adapter is not a bus device. The following discussion walks through the steps required to build the device tree shown in Figure 2. The example begins with the objects at the bottom of the stack.

1. **ACPI HAL PDO and FDO**. The ACPI HAL is detected during Setup, and the root enumerator creates a PDO for the ACPI HAL. The driver creates an FDO, and the ACPI HAL attaches it to the device stack for ACPI (shown by a curved arrow in Figure 2).
2. **ACPI PDO and FDO**. The ACPI HAL enumerates ACPI and creates a PDO for it. The Plug and Play Manager identifies Acpi.sys as the function driver for ACPI, loads the driver, and passes the PDO to the ACPI driver. The ACPI

driver creates an FDO for ACPI and attaches it to the device stack for ACPI.

3. **PCI adapter PDO and FDO.** Plug and Play directs the ACPI driver to enumerate ACPI and create a PDO for each device it finds. In Figure 2, a PCI bus controller is found. The Plug and Play Manager identifies the PCI driver (Pci.sys) as the function driver for the PCI controller, loads the PCI driver, and passes the PDO to the PCI driver. The PCI driver creates an FDO for the PCI bus and attaches the FDO to the device stack for the PCI controller. Creating and attaching this FDO are the PCI driver's responsibilities as the function driver for the PCI controller.

   An IHV can add functionality to a bus by supplying a bus filter driver (not shown in Figure 2). This method is preferred to supplying a custom bus driver.

4. **LAN adapter and USB host controller PDOs and FDOs**. The Plug and Play Manager directs the PCI driver to enumerate the PCI bus. In this example, the PCI bus driver finds a LAN adapter and a USB host controller, and creates PDOs for both devices.

   The Plug and Play Manager identifies the LAN adapter driver as the function driver for the LAN adapter, loads the driver, and passes the adapter PDO to the LAN adapter driver. The LAN adapter driver creates an FDO for the LAN adapter and attaches the FDO to its device stack.

   The Plug and Play Manager identifies the USB host controller driver (either Uhcd.sys or Openhci.sys, depending on the chip set used) as the function driver for the USB host controller, loads the driver, and passes the USB host controller driver the USB host controller PDO. The USB host controller driver creates an FDO for the USB host controller and attaches the FDO to its device stack.

5. **USB hub PDO and FDO.** The Plug and Play Manager directs the USB host controller driver to enumerate the USB host controller. In this example, the USB host controller driver finds a USB hub and creates a PDO for it. The Plug and Play Manager identifies the USB hub driver (Usbhub.sys) as the function driver for the USB hub, loads the driver, and passes the USB hub driver the USB hub PDO. The USB hub driver creates an FDO for the USB hub and attaches the FDO to its device stack.

   The PDO created by the underlying bus driver is always at the bottom of the device stack for a particular device. When drivers handle Plug and Play and power management IRPs, they must pass the IRP all the way down the device stack to the PDO and its associated bus driver.

6. **Re-enumeration of devices as a result of a power state change.** Systems that go to sleep often cannot detect whether a device has been removed or added while the system was asleep because the hardware circuitry that detects such an event may not be available. When an ACPI system wakes up, the ACPI driver can notify an existing bus driver that there is a device check on its

device, providing the signal to the bus driver that it needs to re-enumerate its devices. Good system design provides for selective notification on wake-up rather that requiring the entire system to re-enumerate devices.

The PDO created by the underlying bus driver is always at the bottom of the device stack for a particular device. When a driver handles a Plug and Play IRPs, it must pass the IRP all the way down the device stack to the PDO and its associated bus driver.

CONCLUSION

A major factor in the customer experience for Windows 2000 is the availability of tested, certified Plug and Play device drivers at the time that the operating system ships. The list of supported Windows Plug and Play drivers allows Microsoft to notify customers of hardware issues that they may encounter during the operating system upgrade. This list of supported hardware will also influence the purchasing decisions of both OEMs and enterprise customers. For these reasons, it is vital that writers of device drivers incorporate Plug and Play and power management functionality into their drivers as soon as possible. It is recommended that you update your drivers to include Plug and Play functionality and submit these updated drivers to the Microsoft Windows Hardware Quality Lab (WHQL) for testing and certification.

Look for more information on Plug and Play drivers in the coming Windows 2000 DDK and in future white papers in the Hardware News Letter.

REFERENCES

Advanced Configuration and Power Interface Specification, Version 1.0
http://www.teleport.com/~acpi/

Microsoft DDKs for Windows operating systems, including NDIS documentation
Microsoft Developer Network (MSDN) Professional membership

Note: The Windows 2000 Device Driver Kit documents the Plug and Play interfaces and provides background information on Plug and Play, power management, and the Win32 Driver Model. The Windows 2000 DDK will be substantially updated from the Windows 98 DDK Preview, which provided an early introduction to this material.

"OnNow and ACPI: Introduction and Specifications" and related white papers
Power management specifications for device and bus classes
http://www.microsoft.com/hwdev/onnow.htm

OnNow capabilities and power management
http://www.microsoft.com/hwdev/pcfuture/onnowwdm.htm

PCI Local Bus Specification, Revision 2.1 (PCI 2.1)
http://www.pcisig.com

PCMCIA standards
http://www.pc-card.com

Plug and Play specifications
http://www.microsoft.com/hwdev/specs/

USB Specification, Version 1.0
http://www.usb.org

SUMMARY (or Conclusion)

For More Information

For the latest information on Windows 2000, check out our World Wide Web site at http://www.microsoft.com/ntserver/, the Windows NT Server Forum on MSN™, and The Microsoft Network online service (GO WORD: MSNTS).